VR Props: An End-to-End Pipeline for Transporting Real Objects into Virtual and Augmented Environments

Catherine Taylor*
University of Bath
Marshmallow Laser Feast

Chris Mullany†
Marshmallow Laser Feast

Robin McNicholas‡
Marshmallow Laser Feast

Darren Cosker§
University of Bath

ABSTRACT

Improvements in both software and hardware, as well as an increase in consumer suitable equipment, have resulted in great advances in the fields of virtual and augmented reality. Typically, systems use controllers or hand gestures to interact with virtual objects. However, these motions are often unnatural and diminish the immersion of the experience. Moreover, these approaches offer limited tactile feedback. There does not currently exist a platform to bring an arbitrary physical object into the virtual world without additional peripherals or the use of expensive motion capture systems. Such a system could be used for immersive experiences within the entertainment industry as well as being applied to VR or AR training experiences, in the fields of health and engineering.

We propose an end-to-end pipeline for creating an interactive virtual prop from rigid and non-rigid physical objects. This includes a novel method for tracking the deformations of rigid and non-rigid objects at interactive rates using a single RGBD camera. We scan our physical object and process the point cloud to produce a triangular mesh. A range of possible deformations can be obtained by using a finite element method simulation and these are reduced to a low dimensional basis using principal component analysis. Machine learning approaches, in particular neural networks, have become key tools in computer vision and have been used on a range of tasks. Moreover, there has been an increased trend in training networks on synthetic data. To this end, we use a convolutional neural network, trained on synthetic data, to track the movement and potential deformations of an object in unlabelled RGB images from a single RGBD camera. We demonstrate our results for several objects with different sizes and appearances.

Keywords: Virtual Reality, Real Time Tracking, Virtual Objects, VR Props.

Index Terms: Computing methodologies—Neural networks Computing methodologies—Modelling and simulation Computing methodologies—Computer vision

1 INTRODUCTION

Over recent years, both commercial and academic interest in augmented and virtual reality (VR and AR) has increased greatly as a result of advances in hardware and software. Traditionally, AR and VR were used within the entertainment industry. However, popularity for this medium is growing within medicine, engineering and health. A key aspect of a successful VR or AR application is the feeling of immersion. While many systems use a controller to allow a user to interact with the virtual environment, this may feel unnatural. In addition, virtual training systems which use such controls are limited by how much these motions model real world behaviour. On the other hand, a system which allows a user to transport a real world object into a virtual world - for use as a control mechanism or as a proxy for the real object - may feel more intuitive and so increase immersion. To address this limitation, we develop an efficient end to end pipeline that takes an arbitrary rigid or non-rigid real-world object and transforms this into a virtual object, which we call a *VR Prop*.

Capturing the position, rotation and potential deformations of an arbitrary object from a single view is a challenging task. The appearance of an 3D object in a 2D image is effected by viewpoint, scale and lighting and so reconstruction can be ambiguous [11, 26, 32]. In addition, the shape of a non-rigid object can change greatly as it is deformed by undergoing, bends, twists and stretches. A benefit of tracking an object for a virtual application is that we are able to have full control over the physical environment so can ensure a controlled capture environment (e.g. lighting, green-screen) as well as add markers to objects (if required) to minimise ambiguity.

At present, physical objects can be brought into a virtual environment in several ways, such as using sensors on the surface of the object or by tracking markers. HTC developed a *Vive tracker* which can be attached an object to bring it into a virtual scene [10]. The high accuracy tracking and ability to register feedback from the object, such as button presses, has potential for increased immersion. However, the tracker only computes the position and orientation of the object and so this sensor cannot capture deformations. Alternatively, motion capture systems, such as those by Vicon, can detect points or markers on the surface of an object and use these to drive the motion of a rigged model [36]. This requires a rigged model to be created to represent the physical model. Additionally, motion capture systems require multiple powerful cameras and sensors and so can be a costly solution.

Neural networks are now key components of modern computer vision, with several notable works using these to track objects in RGB images [1, 11, 27, 39]. However, these approaches either require multiple RGB cameras or a substantial amount of labelled training data making the approach difficult to use for new arbitrary objects without a substantial amount of manual training effort.

In our paper, we present a full pipeline which transforms an arbitrary rigid or non-rigid real-world object into an interactive VR prop that can be input to a virtual environment and manipulated by a user - without the need for manual labelling or training. Our pipeline automatically generates a rigged blendshape model of the physical object without any manual artistic input which may require creating new meshes or bones. However, our pipeline is flexible and can also be used with manually created models and rigs should the user require it. Objects are captured with a 3D scanner, and a combination of Finite Element Modelling (FEM) and statistical modelling (i.e. PCA) is used to create the animation model. It has recently been shown that networks trained on synthetic images can still make accurate predictions on real images, allowing networks to be trained on vast amounts of data without having to collect that data by hand [1, 27]. Our pipeline exploits this by training a network with synthetic data from the object such that our CNN can predict object orientation and shape (blendshape weights) from unlabelled RGB images. At run time, the trained network makes predictions of object

*e-mail: c.taylor3@bath.ac.uk
†e-mail: me@chrismullany.com
‡e-mail: robin@marshmallowlaserfeast.com
§e-mail: dpc22@bath.ac.uk

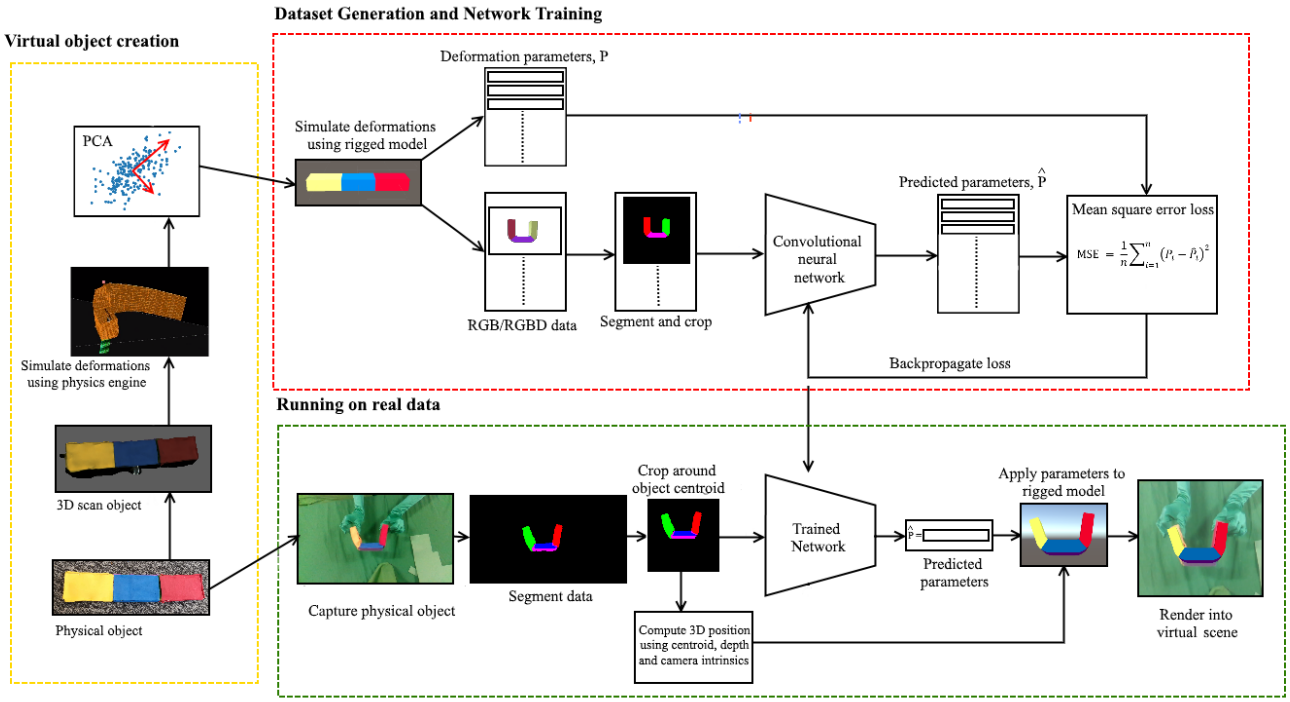


Figure 1: Our proposed end to end pipeline for creating an interactive virtual prop from a physical object. To begin (Section 3.1), a virtual object is created by 3D scanning the physical item. A wide range of deformations are generated using an FEM simulation and these reduced down via PCA into blendshapes. The rigged model is used to build a synthetic dataset and train a CNN to predict deformation parameters from unlabelled RGB images (Section 3.2). Finally, the trained network predicts deformation parameters from RGB images from a single RGBD sensor and these are used to update the pose and shape of the virtual model (Section 3.3).

pose and deformation from RGB images of the physical objects from a single RGBD camera. The predicted parameters update the behaviour of the computer generated model at interactive rates.

Our main contributions from this work are as follows:

1. An end-to-end pipeline for creating an interactive virtual prop from an physical object which can be manipulated in the virtual world, without requiring manual sculpting or rigging.

2. A novel neural network approach for interactive tracking of arbitrary volumetric non-rigid objects in RGB images using synthetic training data without manual labelling.

We continue the paper in Section 2 with a review of the related work. In Section 3, we explain each stage of our end to end pipeline: virtual object creation (Section 3.1), dataset generation and network training (Section 3.2) and running on real data (Section 3.3). We present the results of our pipeline in Section 4 and discuss our approach in Section 5. Finally, in Section 6 we conclude and propose future directions.

2 Related Work

Interacting with Virtual and Augmented Objects: Traditionally, controllers have been used to interact with virtual objects. As a basic case, the computer generated object could react to button clicks on a simple controller or console and different behaviours generated using sequences of button presses. In recent years, controllers have become more sophisticated such that the position and orientation of the controllers can be used alongside buttons to interact with virtual objects. Controllers such as these are used by the HTC Vive [10] and Oculus Rift [22] to create a more immersive experience. HTC [10] offer additional sensors known as Vive Trackers which can be attached to physical objects and used to accurately track their position and orientation. However, these trackers are limited to detecting rigid transformations. Alternatively, Microsoft's HoloLens [19] captures hand gestures and uses these as means of interacting with a computer generated object. While these methods offer increased immersion, they are still limited as they feel unnatural and do not accurately represent the intuitive way to interact with a physical object.

In contrast, the motion of a physical object can be tracked, using an RGB or RGBD camera, and used to control the behaviour of a virtual object. Feature points on an object of interest can be detected and a virtual model or template fit to this data [23, 28, 34]. Augmented Things - a novel approach by *Rambach et al.* [28] - combines the internet of things with 3D object tracking for augmented reality. However, this method is restricted to tracking rigid objects, while our method can capture non-rigid deformations. In their template fitting method, *Tjaden and Schomer* [34] use local colour histograms as feature descriptors to make tracking method robust to occlusions but, again, this approach is limited to rigid motions. Another set of work carries out non-rigid object tracking by fitting a template or model to some observed data [12,14,35]. The parameters of the non-rigid model can be learnt using user labelled data [12]. In contrast, in our method, the deformation and rigid transform parameters can be learnt from unlabelled images. This makes our pipeline more automatic and our network unsupervised as we can quickly obtain large datasets for training without time-consuming manual labelling. *Tsoli et al.* [35] capture complex deformations by jointly tracking the object of interest and the hands which are interacting with it. While this method shows good results using the hand interactions, the authors do not present their work

as a real-time or interactive method and provide no information on frame-rate or efficiency.

Motion capture systems, such as Vicon, can be used to accurately track markers or points on the surface of an object and use these to drive the motion of a rigged model, for example a human skeleton [36, 37], or track a rigid object. The tracked motion can then be used to to control the behaviour of a virtual rigid or non-rigid object. However, these approaches are still limited in that object positions are only sparsely recorded, markers can interfere with a users hands and if moved will affect tracking, and most critically such systems are very expensive requiring non-standard hardware.

Modelling Non-Rigid objects: To create objects which may be deformed in a virtual world, a suitable model is required. Prior work on modelling non-rigid objects can largely be divided into two main categories: using statistical models or physics-based approaches.

In a statistical model, each objects' deformation or shape can be represented as a linear combination of basis vectors [11,17,18,30,31]. The basis vectors can be manually sculpted and used to build a rigged model. As a more efficient alternative, the basis vectors can be automatically generated by analysing a dataset containing different deformations [18, 30, 31]. Principal component analysis (PCA) is a useful technique for calculating a low dimensional representation of a dataset by examining the variation within the data [33]. Statistical models have been shown to be able to accurately model complex objects such as faces, bodies and hands [11, 17, 18, 30]. The skinned multi-person linear model (SMPL) is a prime example of how a object which can go under a wide number of deformations can be modelled using a low number of parameters [18]. *Loper et al.* [18] build their human body model from many 3D scans of different people in a variety of poses and learn parameters such as shape, pose as well as joint angle location.

A contrasting option to model a non-rigid object is using a physics-based model, where the behaviour of an object is controlled by a system of equations. The equations consider the external forces acting upon an object as well as the internal behaviour, due to elasticity, stiffness and willingness to compress. One such approach is a mass spring damper system which are able to model small deformations [7, 32]. However, these do not preserve volume and cannot handle large deformations. On the other hand, the finite element method (FEM) is a commonly used, volume preserving approach which can capture large deformations [5]. It has been used in a number of recent works for tracking of non-rigid objects [24,25]. These approaches both use a physics engine to solve the system of equations.

In our method, we wish to make use of the simplicity of statistical models without having prior access to a large training dataset for each object we will model. Thus, we first use a physics-based method to simulate a range of deformations before then reducing the dimensions of the deformation space through a statistical model.

Capturing Deformations with Neural Networks: Neural networks have become key tools in computer vision and several novel approaches propose using neural networks for predicting rigid and non-rigid object transformations [1, 4, 6, 11, 13, 27, 39]. Additionally, there has been an increasing trend in training networks on purely, or at least partially, synthetic datasets, allowing large datasets to be created for objects which would otherwise have been too time consuming or difficult to capture and providing ground truth pairs for training [1, 4, 20, 27]. Synthetic augmentations, such as blurring, translating and rotating, can be used to warp input data in order to increase the size of the dataset as well as adding in real-world complexities or helping the system become invariant to rotations or translations [20]. Alternatively, software such as Unity, Maya or CAD can be used to generate 3D models, depth maps or images which can be used to train a network [4, 27]. In our pipeline, we use synthetic data as it provides ground truth pairs of blendshape weights and RGB images which would usually only be obtainable through the manual annotation of thousands of images.

Neural networks have been used to learn deformation parameters, such as blendshape weights, as well as global transformations, such as rotations and translations [1, 6, 11]. *Andrychowicz et al.* [1] predicted rigid motion from multiple RGB images using a series of convolutional neural networks. In a similar manner, *Xiang et al.'s* [39] network PoseCNN predicts rigid motion, in this instance from single RGB images. However, these methods do not extend to non-rigid objects. On the other hand, *Kanawaza et al.* [11] propose an end to end pipeline for recovering 3D human meshes from single 2D RGB images. Their network, which is trained on labelled data, contains a discriminator which tests if the predicted parameters belong to a real model. The discriminator is trained using models from the SMPL dataset. Another key approach by *Pumarola et al.* [27] predicts 3D surface meshes from 2D images with a geometry-aware network. Their novel network consists of a detection branch to find the object the mesh in a 2D image, a depth map to predict the 3D positions of the mesh points and a shape branch which combines the results. In the spirit of these methods, we will use a CNN to predict both blendshape weights and object pose from a single RGB image. However, in contrast to *Kanawaza et al.* [11] unlike for the human body, there may not exist a large labelled dataset for the arbitrary object we wish to place in our virtual environment. Thus, we will train on unlabelled data. Additionally, in contrast to the work by *Pumarola et al.* [27] our network is not restricted to surface meshes and instead we find the deformation of volumetric 3D objects. Moreover, our approach runs at an interactive rate, while theirs does not.

3 End-to-End Pipeline

Our end-to-end pipeline, as displayed in Figure 1, is composed of three sections: virtual object creation (Section 3.1), dataset generation and network training (Section 3.2) and running on real data 3.3). We chose our virtual objects to be triangular meshes and model deformations using blendshapes.

The mesh is generated by 3D scanning the physical object and processing the captured point cloud. A FEM simulation is carried out on the mesh, generating a range of shapes which are then reduced down to a set of key deformations using principal component analysis. The principal components which correspond to 90% of the variation in the dataset and saved out as blendshapes at $(\pm)2$ standard deviations (s.d) of the eigenvector.

The dataset is generated by randomly sampling the position, orientation and blendshape weights of the object and rendering the corresponding RGB image. The images are segmented and cropped, ready to train our network with. We train a CNN using the synthetic dataset to predict the deformation parameters from unlabelled RGB images. We use the resnet architecture and begin training from the pretrained weights from imageNet classification [9]. We use the mean square error (MSE) as the loss function and optimise using stochastic gradient descent.

Finally, the trained network is used to make predictions on RGB inputs from a single RGBD camera. The predicted deformation parameters are used to update the virtual object which can then be rendered into a computer generated virtual scene. In the following sections we overview in detail each step in our pipeline. Details of our implementation are given in Section 3.4.

3.1 Virtual Object Creation

A key aspect of our pipeline is the ability to transform an arbitrary physical object into an interactive virtual prop. Thus, we propose creating our virtual representation directly from the physical object, without having an artist having to manually build the chosen object. This allows complex objects to be modelled without requiring 3D sculpting expertise. We begin by 3D scanning our chosen object and processing the captured point cloud to obtain a high resolution

triangular mesh. The scanning process also captures a colour image of the real-world object's appearance, which can be used to texture the triangular mesh. In a similar manner, *Kausch et al.* [14] automatically generate a model using a multi-view camera set up. We chose to use 3D scanning as it provided a texture as well as a triangular mesh which can be used to assist tracking.

Using a 3D scanner, a point cloud or several point clouds are captured for our chosen object. If there are multiple point clouds these must be rigidly aligned and globally registered to a single point cloud, using colour and 3D position information. The colour information ensures that symmetric point clouds are properly aligned. Finally, the unordered point cloud can be fused by Delaunay triangulation to form a polygon mesh [29]. These steps are carried out within the scanner software [2].

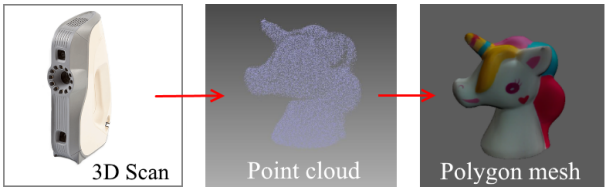


Figure 2: A textured triangular mesh is created from the point clouds captured from the 3D scanner [2]. The point clouds are first aligned and registered globally. They are then fused to form a polygon mesh.

For a rigid object, the 3D triangular mesh is an appropriate virtual representation as the pose can be changed by simply multiplying each vertex by a 4×4 transformation matrix and we do not need to change the shape.

However, for a non-rigid object we must be able to model a range of deformations. We do this using a blendshape model which is learnt from a large range of simulated shapes [16]. A blendshape model was chosen object as deformations are simply represented by a small number of parameters, the blend weights. Additionally, blendshape models have been show to be suitable representations for many non-rigid objects, even complex items such as hands, faces and bodies [17, 18, 30, 31].

The blendshape model is learnt from a dataset of different poses. Similar to the work by *Salzmann et al.* [31], we simulate a range of deformations and reduce these to several key blendshapes using PCA. However, rather than obtaining poses by varying angles in the mesh, we utilise an FEM simulation [5].

To carry out a simulation, we use an FEM mesh representation of our object. The elasticity of the object is controlled by two parameters: the Poisson ratio and Youngs' modulus. The choice of these restrict which deformations can occur. To deform the object, a variety of forces of different magnitude and orientation can be randomly applied to different points on the objects and the resulting deformations saved. This process creates a large dataset of k meshes, $\mathbf{V}_{unaligned} = [\mathbf{v}^1_{unaligned}, ..., \mathbf{v}^k_{unaligned}]$.

Before, reducing the dimension of this dataset using PCA, the meshes must be aligned using colour and point information to create a dataset of k aligned meshes $\mathbf{V}_{aligned} = [\mathbf{v}^1_{aligned}, ..., \mathbf{v}^k_{aligned}]$. The meshes must be aligned so that PCA determines variation in the dataset due to changes in shape and not changes in pose. The addition of colour information is allows symmetrically shaped meshes to correctly orientated.

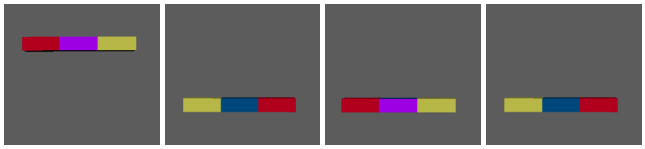
The distance between vertices, $\mathbf{vx}_i = [x_i, y_i, z_i, r_i, g_i, b_i]$, $\mathbf{vx}_j = [x_j, y_j, z_j, r_j, g_j, b_j]$ in a mesh can be defined as follows to include colour distance as well as the euclidean distance.

$$dist(\mathbf{vx}_i, \mathbf{vx}_j) = \alpha||\mathbf{x}_i - \mathbf{x}_j|| + \beta||\mathbf{rgb}_i + \mathbf{rgb}_j|| \quad (1)$$

where $\mathbf{x} = [x, y, z]$ is the 3D position of the vertex and $\mathbf{rgb} = [r, g, b]$ is the colour. The constants α, β are the weights for the euclidean and colour distance respectively. Given a vertex, $\mathbf{vx}_i$, the nearest vertex, $\mathbf{vx}_{nn}$, in a neighbouring mesh is found to be the one which minimises the distance between them

$$\mathbf{vx}_{nn} = \min_{\mathbf{vx}_j} dist(\mathbf{vx}_i, \mathbf{vx}_j). \quad (2)$$

The iterative closest point (ICP) algorithm can be used to partially align a pair of meshes [3]. In the modified version of the algorithm, ICP_{colour}, the nearest neighbour pairs between 2 meshes are found using Equation 2. We add an initial centroid alignment before this to reduce the 3D distance between the objects quickly so that ICP_{colour} does not require as many iterations. Moreover, this step allows us to manually chose an alternative alignment point if required, such as the centroid of a subsection of the object. Figure 3 demonstrates the necessity of using the colour distance alongside the euclidean distance.



(a) Source mesh (b) Goal mesh (c) *ICP* (d) ICP_{colour}

Figure 3: *ICP* vs ICP_{colour}. Without the addition of colour, *ICP* aligns the source mesh to the goal mesh using only translation. This correctly aligns the centroid position but the orientation of the mesh is incorrect. On the other hand, ICP_{colour} translates and rotates mesh and correctly aligns both the position and the orientation.

An overview of our chosen algorithm which aligns the meshes from the FEM simulations is outlined in Algorithm 1.

Algorithm 1 Mesh Alignment

Align a dataset of meshes of the same object with different pose and shape
$\mathbf{v}_{goal} = \mathbf{V}_{unaligned}[0]$
$\mathbf{c}_{goal} = centroid(\mathbf{v}_{goal})$
for each $\mathbf{v}_{unaligned}$ in $\mathbf{V}_{unaligned}$ **do**
    $\mathbf{c}_{src} = centroid(\mathbf{v}_{unaligned})$
    $\mathbf{t} = \mathbf{c}_{src} - \mathbf{c}_{goal}$
    for each vertex, $\mathbf{vx}_i$, in $\mathbf{v}_{unaligned}$ **do**
        $\mathbf{vx}_i += t$
    end for
    $\mathbf{v}_{aligned} = ICP_{colour}(\mathbf{v}_{goal}, \mathbf{v}_{unaligned})$
    $\mathbf{V}_{aligned}[i] = \mathbf{v}_{aligned}$
end for
return $\mathbf{V}_{aligned}$

Finally, PCA is applied to the aligned meshes, $\mathbf{V}_{aligned}$. The principal components which represent 90% of the variation within the training data are used as blendshapes at $\pm$2s.d of the eigenvector. Using the n generated blendshapes, $\mathbf{b} = [\mathbf{b}_1, ..., \mathbf{b}_n]$, a new deformation is created using Equation 3

$$\mathbf{v}_{new} = mean(\mathbf{V}_{aligned}) + \sum_{i=1}^{n} w_i \mathbf{b}_i \quad (3)$$

where $mean(\mathbf{V}_{mean})$ is the mean shape, calculated from the aligned meshes, and $\mathbf{w} = [w_1, ..., w_n]$ is a vector of the blend weights. The weights are between 0 and 1. Using PCA to create blendshapes ensures that the basis vectors are orthogonal. Thus, there will be no interference between blendshapes, which would result in an unnatural deformation being modelled.

3.2 Dataset Generation and Network Training

A synthetic training dataset is generated from the virtual model, $\mathbf{v}$. To create a wide range of poses and deformations, the blendshape weights, $\mathbf{w} = [w_1, ..., w_n]$, 3D position, $\mathbf{T} = [T_x, T_y, T_z]$, and orientation, $\mathbf{R}_{3\times3}$ of the object are varied. Our data generation produces a set of K cropped and segmented RGB images, $\mathbf{I} = [\mathbf{I}^1_{crop}, ..., \mathbf{I}^K_{crop}]$, and the corresponding parameters that have been used to deform the model, $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_K]$. Each deformation parameter is of the form $\mathbf{d} = [w_1, ..., w_n, R_{1,1}, R_{1,2}, R_{1,3}, ..., R_{3,3}]$, where w_i are the blend weights for each of the n blendshapes in the model and $R_{i,j}$ are the entries of the 3×3 rotation matrix, R.

The model can be placed anywhere within the view of the camera. Therefore, $\mathbf{v}$ is moved between frames by randomly selecting the 3D position, $\mathbf{T}$, to be a point within the set $\mathbf{VP} = \{(x,y,z)|(x,y,z) \in$ camera view$\}$. To vary the orientation of the object, the angles of rotation around the x, y, z axes, rot_x, rot_y, rot_z are randomly sampled in the range $(0, 2\pi)$. These can be combined to a single $3x3$ rotation matrix, $\mathbf{R} = \mathbf{R}_x(rot_x)\mathbf{R}_y(rot_y)\mathbf{R}_z(rot_z)$, where $R_i(rot)$ is the 3×3 matrix representing a rotation of rot° around the $i-axis$. Finally, the shape of the object is changed, using Equation 3, by uniformly selecting a value for the blend weight in the range $(0, 1)$. An RGB image is rendered for each frame, as well as recording the deformation parameters. We set the virtual camera parameters, in particular the focal length and field of view, equal to those of the camera we use to capture the real object. This is to ensure that the synthetic images are as close as possible to the captured RGB images. Another assumption in the dataset creation is that the camera position and orientation is fixed. An overview of our dataset generation is given in Algorithm 2.

Algorithm 2 Dataset Generation

Generate a synthetic dataset of an object in a variety of poses and under different deformations.
for each frame in range $(0, K)$ **do**
  Randomly vary deformation paramters:
  $\mathbf{T} \in \mathbf{VP}$
  $\mathbf{R} = \mathbf{R}_x(rot_x)\mathbf{R}_y(rot_y)\mathbf{R}_z(rot_z)$ with $rot_x, rot_y, rot_z \in (0, 2\pi)$
  for blend weight, w_i, in $\mathbf{w}$ **do**
    $w_i \in (0, 1)$
  end for
  $\mathbf{d} = [w_1, ..., w_n, R_{1,1}, R_{1,2}, R_{1,3}, ..., R_{3,3}]$
  Deform v:
  for each vertex $\mathbf{vx}$ in v **do**
    $\mathbf{v} = \mathbf{R}\mathbf{v}$
    $centroid(\mathbf{v}) = \mathbf{T}$
    $\mathbf{v} = mean(\mathbf{v}_{aligned}) + \sum_{i=0} w_i \mathbf{b}_i$
  end for
  Render RGB image
end for

As our approach is for a virtual environment, we can have complete control over the appearance of the object and so we chose to register our model to RGB data [1, 11, 24, 31]. *Petit et al.* [25] offer a contrasting approach and fit the model to a point cloud captured by a depth sensor. However, this method is ambiguous and certain transformations, such as an object being rotated around its axis of symmetry would not be registered. *Leizea et al.* [15] also propose an RGBD solution for tracking non-rigid objects. However, although their registration approach is real-time, the tracking and detection algorithms they use are not, and they suggest people implementing their approach use alternative methods. In contrast, we propose an end-to-end system including tracking and registration that runs at interactive rates. The work by *Newcombe et al.* [21] - DynamicFusion - merges together RGBD scans to estimate a 6D motion field of a scene in real time. While this produces impressive results, DynamicFusion is often unable to recover from model failure and reinitialise tracking. On the other hand, our method makes predictions per frame so can recover from fast motions or the object going out of and then returning to the camera view point.

The RGB images are processed before being input to the network. We segment and flatten the images using simple colour thresholding. The segmentation removes all pixels which do not correspond to object of interest. In the flattening step, we remove all shading and slight colour variation in object sections to make the network uniform to colour and lighting differences between the synthetic training data and the real-world object. To do this, pixels within a similar range are extracted and set to the same colour. Algorithm 3 shows an overview of our algorithm and uses an RGB image, $\mathbf{I}_{input}$, of an object which is made up shades of red, green and blue sections as an example. In general, objects can be coloured the same so that threshold tuning is not required for different objects or setups. The algorithm returns the segmented and flattened image, $\mathbf{I}_{out}$.

Algorithm 3 Segment and Flatten a RGB image

Segment the foreground of an RGB images and flatten the colours using colour thresholding
$[\mathbf{bk}_L, \mathbf{bk}_U]$ = background colour range
$[\mathbf{r}_L, \mathbf{r}_U]$ = red range
$[\mathbf{g}_L, \mathbf{g}_U]$ = green range
$[\mathbf{b}_L, \mathbf{b}_U]$ = blue range
for each row, col in $\mathbf{I}_{in}$ **do**
  if $\mathbf{bk}_L \leq \mathbf{I}_{in}[row, col] \leq \mathbf{bk}_U$ **then**
    $\mathbf{I}_{out}[row, col] = [0, 0, 0]$
  else if $\mathbf{r}_L \leq \mathbf{I}_{in}[row, col] \leq \mathbf{r}_U$ **then**
    $\mathbf{I}_{out}[row, col] = [255, 0, 0]$
  else if $\mathbf{g}_L \leq \mathbf{I}_{in}[row, col] \leq \mathbf{g}_U$ **then**
    $\mathbf{I}_{out}[row, col] = [0, 255, 0]$
  else if $\mathbf{b}_L \leq \mathbf{I}_{in}[row, col] \leq \mathbf{b}_U$ **then**
    $\mathbf{I}_{out}[row, col] = [0, 0, 255]$
  end if
end for
return $\mathbf{I}_{out}$

The input to the network is a square image. Therefore, the segmented and flattened image, $\mathbf{I}_{out}$, must be cropped. The centroid of the object in the 2D image can be calculated using the coordinates, $[(x_1, y_1), ..., (x_N, y_N)]$, of the N non-zero pixels

$$centroid(\mathbf{I}_{out}) = \frac{1}{N}\sum_{i=1}^{N}(x_i, y_i). \quad (4)$$

We select the centroid as the centre of the image and crop around that point. Using the training dataset, a convolutional neural network is trained to predict pose and blendshape weights from unlabelled RGB images. The network takes a square RGB image as an input, $\mathbf{I}_{crop}$, and returns the predicted deformations parameters, $\hat{\mathbf{d}}_i$

$$f(\mathbf{I}_{crop}) = \hat{\mathbf{d}}_{\mathbf{I}_{crop}}. \quad (5)$$

The network is trained using a mean square error (MSE) loss which finds the sum of squared differences between the K ground truth deformation parameters, $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_K]$, and the predicted parameters, $\hat{\mathbf{D}} = [\hat{\mathbf{d}}_1, ..., \hat{\mathbf{d}}_K]$.

$$MSE(\mathbf{D}, \hat{\mathbf{D}}) = \frac{\sum_{i=1}^{K}(\mathbf{d}_i - \hat{\mathbf{d}}_i)^2}{K}. \quad (6)$$

During training, the error from the loss function expressed in Equation 6 is backpropagated through the network to update the parameter weights. The network is trained until this error converges. Details of the implementation are given in Section 3.4.

3.3 Running on Real Data

The trained network is used to make predictions from real-world images and drive the motion of the virtual model at interactive rates. The deformed model can be rendered into a computer generated scene to be used in a VR or AR environment. If this method is for a virtual environment, we can have total control over physical scene. To this end, the object can be placed in front of a green screen which can be easily thresholded out.

The physical object is captured using a single RGBD camera and the RGB image segmented, flattened and cropped around the centroid, as with the synthetic images. The segmented RGB image is used in turn as a mask to segment the depth map. The mean depth, $depth$, of the object is calculated and used alongside the centroid, $\mathbf{c} = [c_x, c_y]$ and the camera parameters to determine the 3D position, $\mathbf{T}$, in space of the object

$$\mathbf{T} = \begin{bmatrix} \frac{depth}{(c_x - u_x)} f_x \\ \frac{depth}{(c_y - u_y)} f_y \\ depth \end{bmatrix} \tag{7}$$

where $\mathbf{f} = [f_x, f_y]$ and $\mathbf{u} = [u_x, u_y]$ are the focal length and principal point of the camera respectively.

The cropped image is input to the trained network and the predicted blendshape weights and orientation returned as demonstrated in Equation 5. The weights, orientation and 3D position update the shape and pose of the model, which is rendered into a virtual environment.

To increase the efficiency of the real world algorithm, we make the assumption that the change in pose and shape between several subsequent frames is small. Thus, a prediction does not have to be made for each captured frame. Instead, we can interpolate between key frames. Spherical linear interpolation is used to interpolate the rotation between frames. The blendshape weights and 3D position can be interpolated using a Kalman filter [38]. As well as increasing the efficiency of the algorithm, the addition of interpolation will smooth the motion between frames.

3.4 Implementation Details

Virtual Object Creation: We scan our physical objects using an Artec Eva Scanner and process the mesh within the Artec 11 Software [2]. In the mesh processing step, background objects are removed from the mesh. A further clean up and mesh simplification, if required, is done within Maya. The scanning also captures an RGB image which an be used as a texture. We use the soft body simulation engine SOFA, to carry out an FEM simulation, with a triangular FEM mesh [8]. The FEM mesh is deformed by randomly exerting forces on the surface of the mesh and the deformed meshes are saved out.

Generating dataset: For each object, we create our training dataset in Unity. The position, orientation and shape are randomly changed as discussed in Section 3.2. The camera is placed at the origin and the camera parameters are set to those of the Intel realsense D435 sensor ($\mathbf{f} = [622.084, 622.154], \mathbf{u} = [426.034, 245.07]$). Like this camera, the RGB image is rendered at a resolution of 848×480. The images are then cropped to a square of 384×384.

Network Training: For the CNN, the network architecture is a resnet34 [9]. We begin training the weights from the network pre-trained on ImageNet classification. We standardise the deformation parameters so that they have a mean of 0 and a standard deviation of 1. We use a MSE loss and train the network until this loss converges or falls below a chosen threshold. We use a stochastic gradient descent optimiser with learning rate $1e-4$ and momentum 0.9. We trained our network on an Alieanware Desktop computer with an NVIDIA GeForce GTX 1070 GPU and an Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz. The network is trained until the loss converges or the change in loss between epochs is negligible. The training times for each object can be seen in Section 4.2.

Running on real data: We use an Intel Realsense D435 depth camera to capture the physical object. Our capture setup is shown in Figure 4.

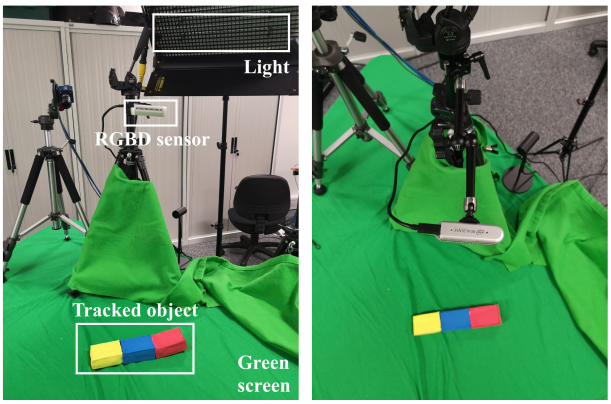


Figure 4: Two views of our setup for tracking rigid motions and deformations from real world objects. The setup consists of an Intel Realsense D435 RGBD sensor which captures our object on a green background. Additionally, a light source can be added to keep the illumination of the object uniform.

The real images are cropped to 384×384, segmented and flattened and input to the network. The network returns the predicted parameters which can be used to update the shape and pose of the virtual object. Finally, the updated computer generated object is rendered into Unity.

4 Experimental Results

To demonstrate our method, we select several rigid and non-rigid objects and create VR props from these. We also look at the success of our tracking method with manually created blendshapes.

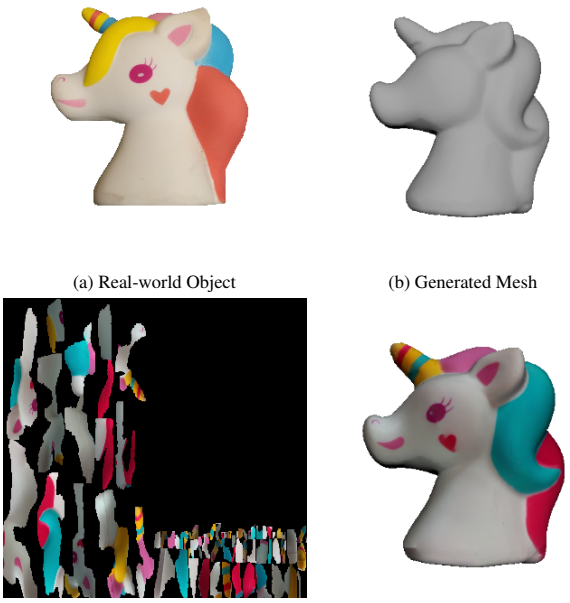
4.1 Virtual Object Creation

We took a number of physical objects and coloured them brightly to aid tracking and reduce 2D-3D projection ambiguities. The object (Figure 5a) is scanned and processed to create a 3D triangular mesh (Figure 5b) and texture image (Figure 5c). These are then combined to create a textured computer generated representation of our chosen object (Figure 5d).

For the rigid object, we chose a box (Figure 6d) to which we attached four different colours: blue, red, yellow and purple. The object itself is symmetric and so the layout of the colours must prevent this symmetry. For the non-rigid objects, we chose a sponge rectangle (Figure 6e) and a unicorn toy (Figure 6f). The sponge object is again coloured non-symmetrically using blue, red, yellow and purple and we use the natural colours of the unicorn.

For a non-rigid object, a large range of deformations are simulated using FEM. In our simulation, we used a triangular FEM mesh. The FEM deformations were reduced using PCA. For our deformable sponge object, we captured around 7000 deformations within SOFA. These were reduced down to 10 blendshapes. A range of poses which can be generated by these blendshapes is demonstrated in Figure 7a. We also created a blendshape model for our unicorn object. This object was created from around 1000 deformations and was reduced to 14 blendshapes. A range of deformations for the unicorn can be seen is Figure 7b.

This approach allows us to capture a wide range of deformations which only depend on a small number of parameters. A high resolution mesh can be deformed in a less costly manner than using



(a) Real-world Object (b) Generated Mesh

(c) UV Texture Map (d) Textured Virtual Object

Figure 5: Creating a virtual representation of a non-rigid coloured unicorn toy from a real-world object. The unicorn is scanned using an Artec Eva Scanner, capturing the shape and texture.

the 3D vertex positions. Additionally, this method allows a 3D computer-generated representation and texture of any chosen object to be easily obtained without requiring the skills of a 3D modeller.

4.2 Tracking Deformations

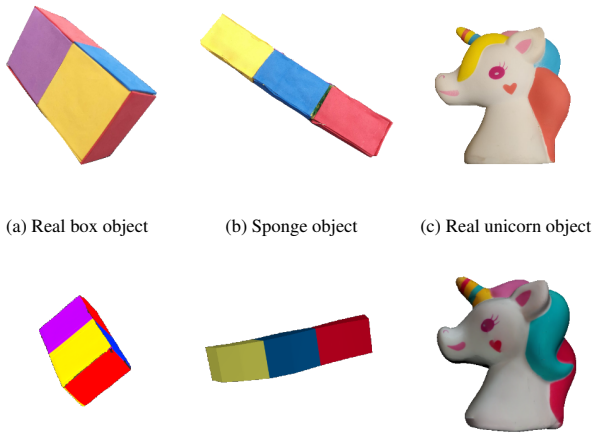
We trained our network separately for each virtual object. We evaluate the success of the training on synthetic and real data.

Synthetic Data: The trained network can be used to predict deformation parameters from a synthetically generated sequence. For each object, we tested our network using a randomly generated dataset of previously unseen data and found the MSE loss per predicted deformation parameter. For each object, the testing error can be seen in Table 1, alongside the training time and the average frame rate for a prediction on synthetic data. The table also shows the size of the training and testing datasets.

	Box	Sponge	Unicorn
Number of Blendshapes	0	2	14
MSE	0.42	0.239	2.37
Training time (hrs)	20	35	35
Prediction rate (fps)	9.98	9.77	9.85
Training dataset size	12459	24951	22592
Testing dataset size	4153	8317	7352

Table 1: MSE per predicted deformation parameter, training time and frame rate for a synthetic prediction for our different objects as well as the size of training and testing datasets.

The results can also be qualitatively analysed. For each object, a synthetic sequence was created, containing a range of poses and, for the non-rigid objects, shapes. For each frame, our trained network predicted the orientation and blendshape weights. These could be



(a) Real box object (b) Sponge object (c) Real unicorn object

(d) Rigid box model (e) Non-rigid sponge model (f) Non-rigid unicorn model

Figure 6: Our chosen objects for testing our end-to-end pipeline for virtual props. There is 1 rigid object and 2 non-rigid objects. Colours are added to the sponge and box object but the natural colours are used for the unicorn. The top row shows the real world objects and the bottom row shows the corresponding computer generated model.

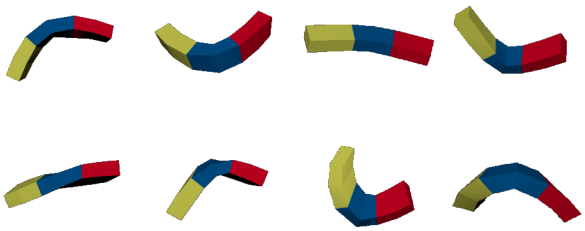
used to update the virtual model. We rendered both the predicted and ground truth objects in Unity so that they could be compared visually. Figure 8 shows the predicted meshes and ground truth meshes for a range of objects which have undergone different transformations. Additionally, each image displays the root mean square (RMS) error, to express how closely the prediction matches the ground truth.

As can be seen in Figure 8, the predicted results closely match the ground truth results. Additionally, the RMS values are relatively low, further confirming the success of our tracking method. The 2 non-rigid objects behave in different ways. The sponge object can undergo larger more articulated deformations which are predicted more accurately than the subtler deformations of the unicorn. As each frame is predicted separately, spherical linear interpolation can be performed on the rotation matrices so that the orientation of the object varies smoothly between frames.

Thus, we have shown that our network is able to learn the blendshape weights and orientation of a range of objects from a single RGB camera. We now demonstrate that our tracking method extends to real-world data, using the network trained on synthetic data.

Real Data: Our network can also be used to predict deformations from real-world data. Figure 9 demonstrates the results of our network on our chosen virtual objects. We show the captured RGB images for each frame and the predicted shape and pose of our virtual object from our system. Further results can be seen in the supplementary videos. To analyse the tracking success, we again use the sponge object with the manually created blendshapes.

Figure 9 shows positive results for a range of poses and deformations. Compelling results are produced for our rigid and non-rigid objects which have very different behaviours. Additionally, accurate predictions have been made for images with partial occlusions, for example from hands. As our system is aimed at VR and AR applications, the time for the prediction from real data is very important. We carryout a prediction in every 2nd frame and interpolate between to increase the efficiency of our algorithm. Taking this account, our algorithm is able to predict deformation parameters from real images at an interactive rate of 15.98fps.



(a) Sponge deformations from blendshape model

(b) Unicorn deformations from blendshape model

Figure 7: A range of deformations generated by randomly varying the blendshape weight in our non-rigid models. The sponge model has 10 blendshapes while the unicorn has 14.

5 Discussion

At present we have used a controlled environment and textured our objects brightly to aid tracking. We make use of the fact that this system is aimed towards virtual reality, where the user will be inside a headset and will not be able to see the physical environment. Thus, we can have complete control over the physical environment without negatively impacting the experience. Several of the key related works are not restricted by these environmental controls but suffer from limitations which our work does not. *Andrychowicz et al.* [1] and *Xiang et al.* [39] use neural networks for object tracking but are not restricted to controlled green screen environments and can handle more naturally textured objects. However, both these approaches are limited to capturing rigid motions. On the other hand, while *Pumarola et al.* [27] capture non-rigid object deformations without texturing or adding markers to the object, their system is not real-time and is limited to surface reconstruction. Finally, *Kanawaza et al.* [11] predict the pose and deformation of 3D meshes but require large amounts of labelled data. In future work, we would like to expand the areas of potential use or our pipeline by making the system suitable for real world environments. We would like simulate different real world complexities such lighting variations, different background and object materials in our dataset generation. This would make our tracking system more robust to colour variations and allow it to be used in less controlled environments.

The objects we test demonstrate the potential of our pipeline and represent many real-world cases of the types of virtual prop a user might require. Its performance on these objects is both fast and accurate enough for VR. However, we acknowledge that this work focuses on simple objects and in future work as well as focusing on accurate capture or more intricate objects of this type - we will also train our network on more dynamic objects (e.g. fabric and other complex articulated objects) and experiment with different representations of non-rigid objects such as articulated models (e.g. bones or physics-based models).

We presented the results of the network which was trained on RGB images. We also experimented with adding depth into our system and training networks on RGBD images. However, we found this did not improve the results and therefore did not include the depth in our training as it was an additional complexity without providing benefits. We acknowledge the benefits depth can potentially add, and using different model types and more complex objects might leverage its benefit more.

6 Conclusion

A system which can create an interactive virtual prop from an arbitrary physical object has the potential to increase the immersion of VR and AR experiences. Moreover, such a system has real-word applications in entertainment and art as well as medicine and engineering. We have proposed the first, to the best of our knowledge, end to end pipeline for producing virtual props from real rigid and non-rigid objects. We leverage the advantages of both physics-based and statistical models in our creation of a virtual object. An FEM simulation is used to generate a wide range of deformations which is reduced to a smaller number of blendshapes using PCA. We create a synthetic dataset to train a CNN to predict deformation parameters from RGB images and use this trained network to carry out real-time predictions. We demonstrate the results of our network on several rigid and non-rigid objects and highlight the success of our method on both real and synthetic data.

References

[1] M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. W. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[2] Artec. 3d object scanner artec eva. `https://www.artec3d.com/portable-3d-scanners/artec-eva`.

[3] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. *Image Vision Comput.*, 10:145–155, 01 1992. doi: 10.1109/ROBOT.1991.132043

[4] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[5] R. D. R. D. Cook. *Concepts and applications of finite element analysis.* Wiley, New York, 3rd ed. ed., 1989.

[6] P. Dou, S. K. Shah, and I. A. Kakadiaris. End-to-end 3d face reconstruction with deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[7] C. Elbrechter, R. Haschke, and H. Ritter. Bi-manual robotic paper manipulation based on real-time marker tracking and physical modelling. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1427–1432, Sep. 2011. doi: 10.1109/IROS.2011.6094742

[8] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, et al. Sofa: A multi-model framework for interactive physical simulation. In *Soft tissue biomechanical modeling for computer assisted surgery*, pp. 283–321. Springer, 2012.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

[10] HTC. Discover virtual reality beyond imagination. `https://www.vive.com/uk/`.

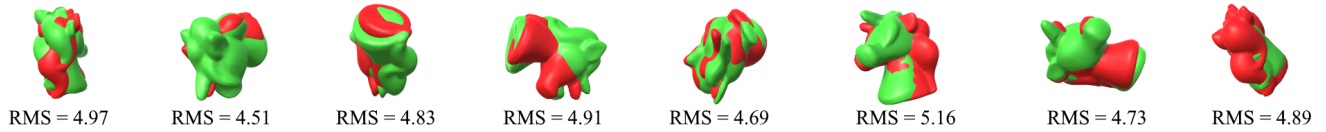
[11] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[12] A. Kanazawa, S. Kovalsky, R. Basri, and D. Jacobs. Learning 3d deformation of animals from 2d images. In *Computer Graphics Forum*, vol. 35, pp. 365–374. Wiley Online Library, 2016.

[13] H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

(a) Predicted vs ground truth on synthetic sponge. The sponge has 2 blend shapes.

(b) Predicted vs ground truth on synthetic box. The box is rigid.



(c) Predicted vs ground truth on synthetic unicorn. The unicorn has 14 blendshapes.

Figure 8: Comparing prediction from synthetic data to ground truth. The green object shows the ground truth under different deformations and orientations and the red shows the corresponding predicted orientation and shape.

[14] L. Kausch, A. Hilsmann, and P. Eisert. Template-based 3d non-rigid shape estimation from monocular image sequences. In *Proceedings of the conference on Vision, Modeling and Visualization*, pp. 37–44. Eurographics Association, 2017.

[15] I. Leizea, H. Álvarez, I. Aguinaga, and D. Borro. Real-time deformation, registration and tracking of solids based on physical simulation. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 165–170. IEEE, 2014.

[16] J. P. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. H. Pighin, and Z. Deng. Practice and theory of blendshape facial models. *Eurographics (State of the Art Reports)*, 1(8):2, 2014.

[17] H. Li, J. Yu, Y. Ye, and C. Bregler. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics (TOG)*, 32, 07 2013. doi: 10.1145/2461912.2462019

[18] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.

[19] Microsoft. Microsoft hololens — mixed reality technology for business. `https://www.microsoft.com/en-us/hololens`.

[20] V. Mondjar-Guerra, S. Garrido-Jurado, R. Muoz-Salinas, M. J. Marn-Jimnez, and R. Medina-Carnicer. Robust identification of fiducial markers in challenging conditions. *Expert Syst. Appl.*, 93(C):336–345, Mar. 2018. doi: 10.1016/j.eswa.2017.10.032

[21] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 343–352, 2015.

[22] Oculus. Oculus rift. `https://www.oculus.com/rift/`.

[23] Y. Park, V. Lepetit, and W. Woo. Multiple 3d object tracking for augmented reality. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 117–120. IEEE Computer Society, 2008.

[24] C. J. Paulus, N. Haouchine, D. Cazier, and S. Cotin. Augmented reality during cutting and tearing of deformable objects. In *2015 IEEE International Symposium on Mixed and Augmented Reality*, pp. 54–59. IEEE, 2015.

[25] A. Petit, V. Lippiello, G. A. Fontanelli, and B. Siciliano. Tracking elastic deformable objects with an rgb-d sensor for a pizza chef robot. *Robotics and Autonomous Systems*, 88:187–201, 2017.

[26] L. Puig and K. Daniilidis. Monocular 3d tracking of deformable surfaces. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 580–586. IEEE, 2016.

[27] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer. Geometry-aware network for non-rigid shape prediction from a single view. 06 2018.

[28] J. Rambach, A. Pagani, and D. Stricker. [poster] augmented things: Enhancing ar applications leveraging the internet of things and universal 3d object tracking. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 103–108. IEEE, 2017.

[29] F. Remondino. From point cloud to surface: the modeling and visualization problem. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34, 2003.

[30] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):245:1–245:17, Nov. 2017.

[31] M. Salzmann, J. Pilet, S. Ilic, and P. Fua. Surface deformation models for nonrigid 3d shape recovery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(8):1481–1487, Aug. 2007. doi: 10.1109/TPAMI.2007.1080

[32] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *2013 IEEE International Conference on Robotics and Automation*, pp. 1130–1137. IEEE, 2013.

[33] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[34] H. Tjaden, U. Schwanecke, and E. Schomer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 124–132, 2017.

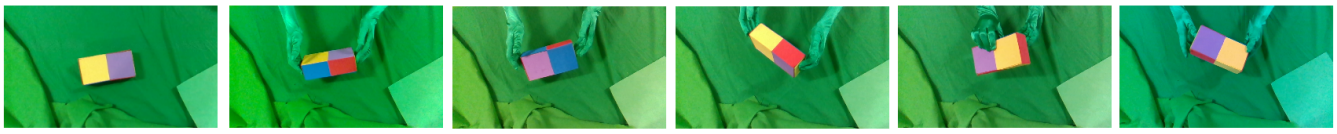
[35] A. Tsoli and A. A. Argyros. Tracking deformable surfaces that undergo topological changes using an rgb-d camera. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 333–341. IEEE, 2016.

[36] Vicon. Motion capture systems. `https://www.vicon.com/`.

[37] Vicon. Origin by vicon. `https://www.vicon.com/press/2018-08-13/origin-by-vicon`.

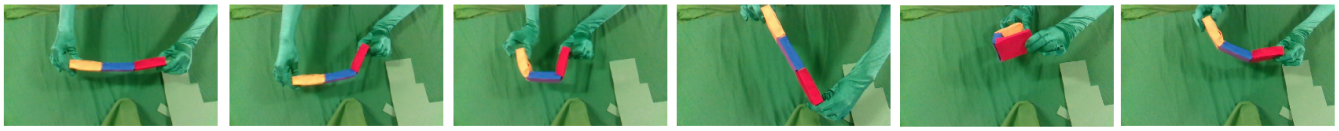
[38] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.

[39] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.



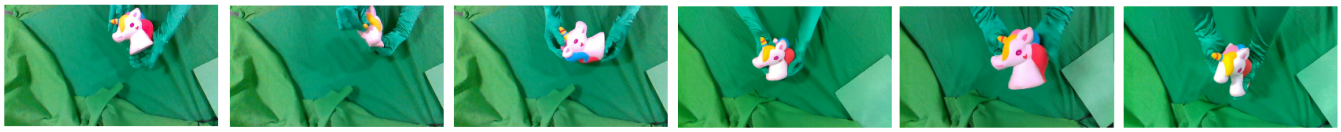
(a) Real world RGB frames of box object

(b) Predicted pose of virtual box



(c) Real world RGB frames of sponge object

(d) Predicted pose and shape of virtual sponge



(e) Real world RGB frames of unicorn object

(f) Predicted pose and shape of virtual unicorn

Figure 9: Shape and pose prediction from real world data. RGB frames of objects changing position and shape are captured and the poses and blendshape weights predicted using our trained CNN.